1. Сервер и клиент

Сервер – это аппаратный или программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам.

Клиент – это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

Клиент и сервер взаимодействуют, используя определённый протокол. Клиент может запрашивать с сервера какие-либо данные, запускать на сервере новые процессы и т. п. Программа-клиент и программа-сервер могут работать как на одном и том же компьютере, так и на разных.

2. База данных

База данных – это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать. Базы данных функционируют под управлением систем управления базами данных (сокращенно СУБД).

3. API

API (Application Programming Interface) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой (напр., клиент взаимодействует с сервером).

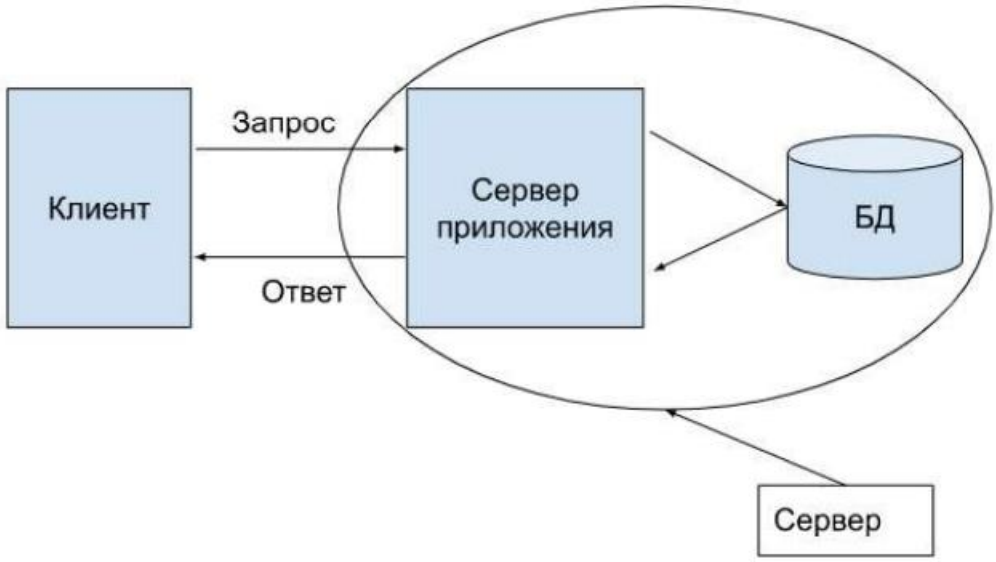
4. Сервис, отличия от сервера

Сервисы – разделенные программные компоненты, предоставляющие определенную функциональность и используемые в составе многих приложений.

Отличия от сервера: Например, веб-сервер – это сервер, реализующий http протокол. Веб-сервис – это технология для взаимодействия между системами.

5. Архитектура клиент-сервер

Модель клиент-сервер является методом распределенных вычислений. Клиент-сервер реализует идею разделения системы на отдельные задачи, размещаемые на различных платформах для большей эффективности.



В клиент-сервере управление данными осуществляется на серверном узле, а другим узлам (клиентам) предоставляется доступ к данным.

Достоинства:

a) основная нагрузка ложится на сервер(а)

b) данные находятся в безопасности, так как сервер дает клиенту только требуемую выборку данных для клиента, основываясь на его уровне доступа

c) разграничение полномочий между клиентом и сервером

d) кроссплатформенность - реализаций клиента может быть сколько угодно: от веб-браузера до приложений мобильных платформ

Недостатки:

a) неработоспособность сервера может сделать неработоспособной всю информационную систему

b) поддержка работы данной системы требует отдельного специалиста

c) высокая стоимость серверного оборудования.

6. Виды сервисов:

a) **Сервер приложений** (англ. application server) – это программная платформа, предназначенная для эффективного исполнения процедур, на которых построены приложения.

b) **Веб-сервер** – сервер, принимающий HTTP-запросы от клиентов, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом или другими данными.

c) **Серверы баз данных** используются для обработки запросов. На сервере находится СУБД для управления БД и ответов на запросы.

d) **Файл-сервер** хранит информацию в виде файлов и предоставляет пользователям доступ к ней.

e) **Прокси-сервер** – промежуточный сервер.

f) **Файрволы** – межсетевые экраны, анализирующие и фильтрующие сетевой трафик, с целью обеспечения безопасности сети.

g) **Почтовые серверы** – предоставляют услуги по отправке и получению электронных почтовых сообщений.

7. **Масштабируемость** – способность работать с увеличенной нагрузкой путем наращивания ресурсов без фундаментальной перестройки архитектуры при добавлении ресурсов. Система называется масштабируемой, если она способна увеличивать производительность пропорционально дополнительным ресурсам. **Вертикальная масштабируемость** представляет собой увеличение производительности компонентов серверной системы в интересах повышения производительности всей системы. Пример: увеличение оперативной памяти, установка более мощного процессора. **Горизонтальная масштабируемость** представляет собой как разбиение системы на более мелкие компоненты и разнесение их, так и увеличение количества компонентов, выполняющих одну и ту же функцию. Пример: добавление еще одного сервера тех же характеристик.

8. **Протокол передачи данных** – набор определенных правил или соглашений, который определяет обмен данными между различными

программами. Эти правила задают единообразный способ передачи сообщений и обработки ошибок.

9. Толстый и тонкий клиенты

При применении **«толстого» клиента** полная функциональность приложения обеспечивается вне зависимости от сервера. В данном случае сервер чаще всего выступает в роли хранилища информации, а вся логика приложения реализуется на клиенте. Даже при отсутствии соединения с сервером работа ведется с локальными копиями данных, а при возобновлении соединения происходит синхронизация данных.

«Тонкий» клиент в отличие от толстого только отображает данные, принятые от сервера. Вся логика приложения выполняется на более производительном сервере, что не требует клиентских мощностей, кроме надежного канала связи.

10. Паттерн MVC: общие тезисы

MVC – фундаментальный шаблон проектирования.

Модель (**Model**) – данные, методы для работы с данными, изменения и обновления данных.

Представление (**View**) – это отображение данных, получаемых от модели, их оформление и другие аспекты презентации модели.

Контроллер (**Controller**) – некий буфер между моделью и представлением. Реагирует на действия пользователя, интерпретирует данные, введенные пользователем, информирует модель и производит необходимые манипуляции с моделью и представлением. В веб-приложениях контроллер управляет запросами пользователя.

11. **Паттерн MVC: Model-View-Presenter**. Особенностью паттерна является то, что он позволяет создавать абстракцию (интерфейс) представления, а презентер получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу меняет модель.

- двусторонняя коммуникация с представлением
- представление взаимодействует напрямую с презентером

- одному презентеру соответствует одно отображение

12. **Паттерн MVC: Model-View-View Model**

Особенностью паттерна является связывание (bindings) – изменение состояния View-модели автоматически изменяет представление и наоборот.

- Двусторонняя коммуникация с представлением.
- View-модель — это абстракция представления. Свойства представления совпадают со свойствами View-модели/модели.
- Одной View-модели соответствует одно отображение.

13. **Docker** – это ПО для развёртывание приложений средствами контейнеризации. Подобно виртуальной машине докер запускает свои процессы в собственной, заранее настроенной операционной системе. Но при этом все процессы докера работают на физическом host сервере, деля всю доступную память с процессами на хосте. Сначала создается docker image (докер-образ) – это тиражируемый образ некоторой «машины», шаблон для создания контейнера – это уже сама «машина», которую можно запускать и останавливать.

14. **Dockerfile** – это исполняемый файл, который собирает образ. Пример наипростейшего заполнения докерфайла: FROM ubuntu:18.04 – данная команда загружает образ ubuntu версии 18.04 с хранилища dockerhub. Другие команды: ADD, COPY, EXPOSE, VOLUME и др.

15. **Docker Compose** – это средство для создания приложения Docker, состоящего из нескольких контейнеров. docker-compose up запустит ваше приложение.

16. **LAMP** — акроним, обозначающий набор (комплекс) серверного программного обеспечения, широко используемый в интернете. LAMP назван по первым буквам входящих в его состав компонентов:

Linux – операционная система Linux

Apache – веб-сервер

MariaDB / MySQL – СУБД

PHP – язык программирования, используемый для создания веб-приложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

17. Конфигурационный файл php.ini

Файл конфигурации php.ini — это текстовый файл с настройками PHP. Содержит набор директив, каждая из которых представляет собой пару ключ-значение. Директивы определяют множество настроек интерпретатора PHP, а также отвечают за загрузку модулей. Возможно настроить языковые опции, ограничение ресурсов и т. п. Строки, начинающиеся с «;», являются комментариями. Для версий серверных модулей PHP считывание php.ini происходит только один раз при запуске веб-сервера.

18. Как написать простой скрипт на php

Установить php и настроить конфигурационный файл php.ini. В тестовом файле с расширением . php в тэге <?php?> можно писать функции, html-тэги и с помощью команды echo выводить текст. Если файл содержит только код PHP, предпочтительно опустить закрывающий тег в конце файла.

19. Основные правила, связанные с переменными в php

- Любая переменная начинается со знака «$»
- Имя переменной может содержать любые буквенные символы, цифры и знак подчеркивания
- Имя переменной может начинаться только с буквы
- Имена переменных чувствительны к регистру
- Переменная инициализируется при помощи знака «=»
- Неинициализированные переменные принимают значение по умолчанию в зависимости от их типа, который определяется из контекста их первого использования
- $this — это специальная переменная, которой нельзя ничего присваивать
- При присвоении по ссылке новая переменная ссылается на оригинальную переменную

20. Основные типы данных в php

PHP поддерживает десять простых типов:

- Четыре скалярных типа: bool, int, float (double), string.
- Четыре смешанных типа: array, object, callable, iterable.
- Два специальных типа: resource, NULL.

21. Какие существуют функции для работы с переменными в php вне зависимости от типа данных

- Boolval/doubleval/floatval/intval и др. — Возвращает логическое значение переменной/ значение переменной в виде числа с плавающей точкой/целочисленное значение переменной и др.
- is_array/is_bool/is_int и др. — Определяет, является ли переменная массивом/булевой/целым числом и др.
- is_numeric — Проверяет, является ли переменная числом или строкой, содержащей число
- is_scalar — Проверяет, является ли переменная скалярным значением
- empty — Проверяет, пуста ли переменная
- get_debug_type — Возвращает имя типа переменной в виде, подходящем для отладки
- gettype — Возвращает тип переменной
- settype — Задаёт тип переменной
- isset — Определяет, была ли установлена переменная значением, отличным от null
- print_r — Выводит удобочитаемую информацию о переменной
- unset — Удаляет переменную
- var_dump — Выводит информацию о переменной

22. Предопределенные переменные в php

Любому запускаемому скрипту PHP предоставляет большое количество предопределённых переменных. Например, суперглобальные

переменные — встроенные переменные, которые всегда доступны во всех областях.

- $GLOBALS — Ссылки на все переменные глобальной области видимости
- $_SERVER — Информация о сервере
- $_GET — Переменные HTTP GET
- $_POST — Переменные HTTP POST
- $_FILES — Переменные файлов, загруженных по HTTP
- $_REQUEST — Переменные HTTP-запроса
- $_SESSION — Переменные сессии
- $_ENV — Переменные окружения
- $_COOKIE — HTTP Cookies
- $php_errormsg — Сообщение об ошибке
- $argc — Количество аргументов, переданных скрипту
- $argv — Массив переданных скрипту аргументов

23. Переменные переменных в php

Переменная переменной берет значение переменной и рассматривает его как имя переменной. Например:

```
<?php
$a = 'hello';
$$a = 'world';
?>
```

Таким образом, выражение `echo "$a ${$a}";` выведет то же, что и `echo "$a $hello";` то есть, они оба выведут: hello world.

24. Выражения в php

PHP — это язык, ориентированный на выражения и рассматривающий почти все как выражение. Самое простое и точное определение выражения - "всё, что имеет значение".

Основными формами выражений являются константы и переменные. Например, $a = 5. 5 - это выражение со значением 5. $a это также выражение со значением 5.

Более сложными примерами выражений являются функции – выражения, значением которых является то, что возвращает функция:

Также выражениями являются префиксный и постфиксный инкремент и декремент, сравнение, а также тернарный условный оператор.

25. Арифметические операторы в php

Пример	**Название**	**Результат**
+$a	Идентичность	Конвертация *$a* в int или float
-$a	Отрицание	Смена знака *$a*.
$a + $b	Сложение	Сумма *$a* и *$b*.
$a - $b	Вычитание	Разность *$a* и *$b*.
$a * $b	Умножение	Произведение *$a* и *$b*.
$a / $b	Деление	Частное от деления *$a* на *$b*.
$a % $b	Деление по модулю	Целочисленный остаток от деления *$a* на *$b*.
$a ** $b	Возведение в степень	Возведение *$a* в степень *$b*.

Результат операции остатка от деления % будет иметь тот же знак, что и делимое.

26. Битовые операции php

Пример	**Название**	**Результат**
$a & $b	И	Устанавливаются только те биты, которые установлены и в *$a*, и в *$b*.
$a | $b	Или	Устанавливаются те биты, которые установлены в *$a* или в *$b*.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в *$a*, либо только в *$b*, но не в обоих одновременно.

~ $a	Отрицание	Устанавливаются те биты, которые не установлены в *$a*, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной *$a* сдвигаются на *$b* позиций влево (каждая позиция подразумевает "умножение на 2")
$a >> $b	Сдвиг вправо	Все биты переменной *$a* сдвигаются на *$b* позиций вправо (каждая позиция подразумевает "деление на 2")

Биты, сдвинутые за границы числа, отбрасываются. Сдвиг влево дополняет число нулями справа, сдвигая в то же время знаковый бит числа влево, что означает что знак операнда не сохраняется. Сдвиг вправо сохраняет копию сдвинутого знакового бита слева, что означает что знак операнда сохраняется.

27. Оператор присваивания в php

Базовый оператор присваивания обозначается как «=». Оператор присваивания означает, что левый операнд получает значение правого выражения. Результатом выполнения оператора присваивания является само присвоенное значение.

В дополнение к базовому оператору присваивания имеются «комбинированные операторы».

```
$b = "Привет";
$b .= "-привет!"; // устанавливает $b в "Привет-привет!", как и  $b = $b . "-привет!";
```

Операторы арифметического присваивания

Пример	**Эквивалент**	**Операция**
$a += $b	$a = $a + $b	Сложение
$a -= $b	$a = $a - $b	Вычитание
$a *= $b	$a = $a * $b	Умножение
$a /= $b	$a = $a / $b	Деление

$a %= $b	$a = $a % $b	Модуль
$a **= $b	$a = $a ** $b	Возведение в степень

Операторы побитового присваивания

Пример	**Эквивалент**	**Операция**
$a &= $b	$a = $a & $b	Побитовое И
$a |= $b	$a = $a | $b	Побитовое ИЛИ
$a ^= $b	$a = $a ^ $b	Побитовое исключающее ИЛИ (Xor)
$a <<= $b	$a = $a << $b	Побитовый сдвиг влево
$a >>= $b	$a = $a >> $b	Побитовый сдвиг вправо

28. Операторы сравнения в php

Пример	**Название**	**Результат**
$a == $b	Равно	**true** если *$a* равно *$b* после преобразования типов.
$a === $b	Тождественно равно	**true** если *$a* равно *$b* и имеет тот же тип.
$a != $b	Не равно	**true** если *$a* не равно *$b* после преобразования типов.
$a <> $b	Не равно	**true** если *$a* не равно *$b* после преобразования типов.
$a !== $b	Тождественно не равно	**true** если *$a* не равно *$b*, или они разных типов.
$a < $b	Меньше	**true** если *$a* строго меньше *$b*.
$a > $b	Больше	**true** если *$a* строго больше *$b*.
$a <= $b	Меньше или равно	**true** если *$a* меньше или равно *$b*.
$a >= $b	Больше или равно	**true** если *$a* больше или равно *$b*.
$a <=> $b	Космический	Число типа int меньше, больше или

	корабль (spaceship)	равное нулю, когда *$a* соответственно меньше, больше или равно *$b*.

В случае если оба операнда являются строками, содержащими числа, или один операнд является числом, а другой - строкой, содержащей числа, то сравнение выполняется численно. Преобразование типа не происходит при сравнении === или !==, поскольку это включает сравнение типа, а также значения.

29. Логические операторы в PHP:

Конструкция **if** предоставляет возможность условного выполнения фрагментов кода.

if (выражение)

инструкция

Выражение вычисляется в булево значение. Если выражение принимает значение true, PHP выполнит инструкцию, а если оно принимает значение false – проигнорирует.

```
<?php
if ($a > $b)
  echo "a больше b";
?>
```

Часто необходимо, чтобы условно выполнялось более одной инструкции.

```
<?php
if ($a > $b) {
  echo "a больше b";
  $b = $a;
}
?>
```

Допускается вложение операторов if.

Блок **else** содержит инструкции, которые выполняются, если условие после if ложно, то есть равно false:

```php
<?php
$a = 4;
if($a > 0){
    echo "Переменная а больше нуля";
}
else{
    echo "Переменная а меньше нуля";
}
echo "<br>конец выполнения программы";
?>
```

Конструкция **elseif** есть сочетание if и else. Аналогично else, она расширяет оператор if для выполнения различных выражений в случае, когда условие начального оператора if эквивалентно false. Однако, в отличие от else, выполнение альтернативного выражения произойдёт только тогда, когда условие оператора elseif будет являться равным true.

```php
<?php
if ($a > $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";
}
?>
```

Тернарная операция состоит из трех операндов и имеет следующее определение: **[первый операнд - условие] ? [второй операнд] : [третий операнд]**. В зависимости от условия тернарная операция возвращает второй или третий операнд: если условие равно true, то возвращается второй операнд; если условие равно false, то третий.

```php
$a = 1;
$b = 2;
$z = $a < $b ? $a + $b : $a - $b;
echo $z;
```

30. Логические операторы в PHP:

Логические операторы

Пример	Название	Результат
$a and $b	И	**true**, если и *$a*, и *$b* **true**.
$a or $b	Или	**true**, если или *$a*, или *$b* **true**.
$a xor $b	Исключающее или	**true**, если *$a*, или *$b* **true**, но не оба.
! $a	Отрицание	**true**, если *$a* не **true**.
$a && $b	И	**true**, если и *$a*, и *$b* **true**.
$a || $b	Или	**true**, если или *$a*, или *$b* **true**.

Смысл двух разных вариантов для операторов "and" и "or" в том, что они работают с различными приоритетами.

PHP Выделить код

```php
<?php
$v = true && false;
var_dump($v); // false

$v = true and false;
var_dump($v); // true
?>
```

У `and` приоритет ниже, чем у оператора присваивания. Т.е. сначала выполнится `$v = true`, а затем `and false`, который уже ничего не изменит.

31. Циклы в php

В цикле **while** вложенные выражения выполняются повторно до тех пор, пока выражение в самом while является true. Значение выражения проверяется каждый раз перед началом цикла, поэтому даже если значение выражения изменится в процессе выполнения вложенных выражений в цикле, выполнение не прекратится до конца итерации. Если выражение while равно false с самого начала, вложенные выражения ни разу не будут выполнены.

```
while (expr)
    statement
```

Цикл **do-while** очень похож на цикл while, с тем отличием, что истинность выражения проверяется в конце итерации, а не в начале. Главное отличие от обычного цикла while в том, что первая итерация цикла do-while гарантированно выполнится.

```
<?php
    $i = 0;
    do {
        echo $i;
```

```
    } while ($i > 0);
?>
```

Синтаксис цикла **for** следующий:

```
for (expr1; expr2; expr3)
   statement
```

Первое выражение (expr1) всегда вычисляется (выполняется) только один раз в начале цикла. В начале каждой итерации оценивается выражение expr2. Если оно принимает значение true, то цикл продолжается и выполняются вложенные операторы. Если оно принимает значение false, выполнение цикла заканчивается. В конце каждой итерации выражение expr3 вычисляется (выполняется).

32. Конструкции switch и match в php

Оператор **switch** похож на ряд операторов IF с одинаковым условием. Данный оператор удобен в том случае, если надо сравнивать одну и ту же переменную с множеством различных значений и выполнять различные участки кода в зависимости от того, какое значение принимает эта переменная.

Оператор switch исполняет строчка за строчкой. Только в случае нахождения оператора case, значение которого совпадает со значением выражения в операторе switch, PHP начинает исполнять операторы. PHP продолжает исполнять операторы до конца блока switch либо до тех пор, пока не встретит оператор break.

```
switch ($i) {
   case 0:
      echo "i равно 0";
      break;
   case 1:
      echo "i равно 1";
      break;
}
```

Выражение **match** предназначено для ветвления потока исполнения на основании проверки совпадения значения с заданным условием. Аналогично оператору switch, выражение match принимает на вход выражение, которое сравнивается с множеством альтернатив. Выполнение кода условий происходит лениво, т.е. код следующего условия выполняется только если все предыдущие проверки провалились. Будет выполнена только одна ветвь кода, соответствующая подошедшему условию.

Отличия от switch:

1. В match используется строгое сравнение (===).
2. Выражение match возвращает результат.
3. В match исполняется только одна, первая подошедшая, ветвь кода, тогда как в switch происходит сквозное исполнение, начиная с подошедшего условия и до первого встретившегося оператора break.

```
<?php
    $food = 'cake';
    $return_value = match ($food) {
       'apple' => 'На столе лежит яблоко',
       'banana' => 'На столе лежит банан',
       'cake' => 'На столе стоит торт',
    };
    var_dump($return_value);
?>
```

33. Include и require в php

Это функции, которые используются для помещения содержимого файла, содержащего исходный код PHP, в другой файл PHP. Разница в том, что функция **include()** выдает предупреждение, но скрипт продолжит выполнение, а функция **require()** выдает предупреждение и фатальную ошибку, т.е. скрипт не будет продолжать выполнение.

34. Функции в php

Функция — очень мощный инструмент повторного использования кода. Создав свою функцию и записав туда необходимый код, вы сможете вызывать и использовать его столько раз, сколько необходимо. В противном случае пришлось бы копировать и вставлять фрагмент кода каждый раз, когда он понадобится.

Разделяют два типа функций — встроенные и пользовательские.

Встроенные функции — это функции, которые за нас уже написали создатели языка программирования, и мы можем просто брать их и использовать. Например, print().

Пользовательские функции программист создает самостоятельно.

Как и с обычными переменными, работа с функциями состоит из их объявления и использования.

```
function <имя функции>(<аргумент функции>){
    <тело функции>
    return <результат выполнения функции>;
}
```

Аргументы функции — это переменные, которые функция может получить из внешнего кода.

Аргументы необходимы, так как функция «не видит» переменные, определённые за её границами. Верно и обратное — переменные, определённые внутри функции, не будут доступны извне.

В отличие от аргументов, которых может быть несколько, вернуть во внешний код функция может только одно значение — с помощью инструкции «return».

36. Что такое веб-сервер?

Веб-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом или другими данными.

37. Что такое сервер приложения и чем он отличается от веб-сервера?

Сервер приложений — это программная платформа, предназначенная для эффективного исполнения процедур, на которых построены приложения.

Основное различие между веб-сервером и сервером приложений заключается в том, что веб-сервер предназначен для обслуживания статических страниц, например HTML и CSS, тогда как сервер приложений отвечает за генерацию динамического содержимого.

38. Кратко опишите историю развития интернета в рамках развития веб-серверов.

Современный интернет начала развиваться в конце 80-ых и начале 90-ых годов прошлого столетия. В то время интернет-сайты были просто хранилищем документов, которые имели специальную разметку. Также сайты сохраняли некоторые связанные с документом данные, например, файлы, изображения и т.д. Для создания ссылок между документом и связанными файлами, был предложен специальный язык гипертекстовой разметки – HTML, а для доступа к таким документам по сети – протокол HTTP (HyperText Transfer Protocol). В 1999 году при смене протокола HTTP/1.1 на более совершенный HTTP/2 произошли некоторые кардинальные изменения, которые были необходимы для развития сети. По мере роста сети Интернет, возможности статического HTML перестали удовлетворять интернет сообщество. От статических web-страниц, трудных в изменении, мы пришли в web-серверам, способным работать с динамическими данными.

39. Кратко опишите протокол HTTP.

HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в Интернете. HTTP является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем (клиентом, обычно веб-браузером). Целью запроса служит некий ресурс, находящийся по определенному URL.

40. Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.

Когда пользователь хочет перейти на страницу, браузер отправляет HTTP-запрос GET с указанием необходимого URL-адреса HTML-страницы. Сервер извлекает запрошенный документ из своей файловой системы и возвращает HTTP-ответ, содержащий документ и код состояния «200 OK» (успех). Сервер может вернуть другой код состояния, например, «404 Not Found», если файл отсутствует на сервере или «301 Moved Permanently», если файл существует, но был перемещён в другое место.

41. Опишите цели и задачи веб-сервера.

Цель веб-сервера – обслуживать одновременно большое количество клиентов, максимально эффективно используя hardware. Главная задача веб сервера – принимать HTTP-запросы от пользователей, обрабатывать их, переводить в цифровой компьютерный код. Затем выдавать HTTP-ответы, преобразуя их из миллионов нолей и единичек в изображения, буквы, HTML-страницы.

42. Опишите технологию SSI.

SSI (Server Side Includes — включения на стороне сервера) – это простая и удобная технология организации, создания и сборки веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа. SSI экономит место на сервере, и одновременно делает администрирование сайта в десятки раз удобнее.

43. Что такое система управления контентом?

Система управления контентом (CMS) — это программное обеспечение, которое работает в вашем браузере. Оно позволяет создавать, управлять веб-сайтом и изменять его содержимое, не имея никаких знаний в области программирования. Система управления контентом предоставляет вам графический интерфейс пользователя, в котором можно управлять всеми аспектами вашего сайта.

44. Верно ли, что сервер приложения умеет работать с протоколом HTTP?

Верно.

HTTP-клиентами чаще всего являются браузеры — Google Chrome, Yandex и др. А серверами являются веб-сервера. Вот эта приставка «веб-» и указывает на то, что это сервер, который умеет принимать запросы и отвечать на них по протоколу HTTP.

45. Что такое CGI?

CGI (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, называют шлюзом.

46. Как работает система с использованием интерфейс шлюза - CGI?

Для передачи данных используются стандартные потоки ввода-вывода, от веб-сервера к CGI-приложению данные передаются через поток STDIN, принимаются обратно через поток STDOUT, а для передачи сообщений об ошибках используется поток STDERR.

47. Назовите достоинства и недостатки CGI.

• CGI не налагает особых условий на платформу и веб-сервер

• Технология не привязана к конкретному языку программирования и может быть использована на любом языке, работающем со стандартными потоками ввода/вывода.

• Производительность CGI-программ не высока. Основной причиной этого является то, что при очередном обращении к серверу для работы CGI – программы создается отдельный процесс, что требует большого количества системных ресурсов.

• Встроенных средств масштабируемости технология не предусматривает.

48. Что такое FastCGI?

FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным.

49. Назовите основные отличия CGI от FastCGI.

При очередном обращении к серверу для работы CGI-программы создается отдельный процесс.

В FastCGI-программе используется очередь запросов, которые обрабатываются последовательно.

50. Что такое менеджер процессов?

Поток process_manager отвечает за генерацию новых экземпляров процесса, за связь со средой и между экземплярами процесса, а также за выполнение экземпляров процесса.

51. Что такое PHP-FPM?

PHP-FPM — это альтернативная реализация FastCGI с несколькими дополнительными возможностями, которые обычно используются для высоконагруженных сайтов.

52. Что такое Spawn-fcgi?

spawn-fcgi — одна из составных частей проекта Lighttpd. Предназначена для того, что бы запустить php как FastCGI сервер.

53. Что такое Lighttpd?

Lighttpd – это веб-сервер с открытым исходным кодом, отвечает за предоставление доступа к статическому контенту через протоколы HTTP или HTTPS.

54. Что такое chroot окружение?

Chroot окружение — системный вызов и просто команда, которая временно перемещает root каталог в новую папку.

Chroot также используется для создания jail и изоляции процессов. Команда выполняется только от имени суперпользователя.

55. Опишите механизм взаимодействия сервисов с использованием FastCGI

Связь между web-сервером и FastCGI-процессом осуществляется через один сокет, который процесс должен слушать на предмет входящих подключений от web-сервера.

web-сервер и FastCGI-процесс обменивается данными с использованием простого протокола, решающего две задачи: организация двунаправленного обмена в рамках одного соединения и организация нескольких независимых FastCGI-сессий в рамках одного соединения.

56. Опишите процесс выбора встроенного или внешнего менеджера процессов.

Встроенный менеджер процессов – удобная программа, с помощью которой можно определять запущенные на ПК приложения, процессы и службы, а также управлять вышеперечисленными. С помощью нее можно оценивать быстродействие компьютера и влиять на него.

К сожалению, встроенный менеджер процессов не всегда удобен для анализа и обработки запущенных процессов и имеет очень ограниченные возможности, кроме того он не в полной мере отображает запущенные процессы. Тогда приходится прибегать к внешним менеджерам процессов. Они более качественно отображают информацию и позволяют лучше контролировать процессы.

57. Что такое SCGI?

SCGI (Simple Common Gateway Interface) - простой общий интерфейс шлюза - разработан как альтернатива CGI и во многом аналогичен FastCGI, но более прост в реализации.

58. Что такое PCGI?

PCGI (Perl Common Gateway Interface) — библиотека к языку программирования Perl для работы с интерфейсом CGI. Основное достоинство заключается в том, что библиотека позволяет совершенно безопасно принимать сколь угодно крупные объёмы данных, при этом очень экономично потребляя оперативную память.

59. Что такое PSGI?

PSGI (Perl Web Server Gateway Interface) – технология взаимодействия веб-сервера и сервера приложений Perl. PSGI-сервер представляет среду для выполнения Perl-приложений, которая постоянно запущена в качестве службы и может взаимодействовать с веб-сервером.

60. Что такое WSGI?

WSGI (Web Server Gateway Interface) – технология взаимодействия веб-сервера и сервера приложений Python.

61. Опишите механизм взаимодействия серверов Apache и PHP

Когда компьютер обращается к web-серверу Apache, то он запускает интерпретатор PHP. Он выполняет скрипт, записанный в файле index.php. То есть Apache – это сервер, который взаимодействует с клиентом (принимает и отвечает на его запросы), а затем сервер выполняет запрос клиента, используя PHP.

62. Опишите преимущества веб-сервера Apache.

Надежность, безопасность и гибкость настройки (Apache позволяет подключать различные модули, добавляющие в него новые возможности). Также к достоинствам Apache необходимо отнести регулярно выпускаемые обновления и патчи.

63. Опишите недостатки веб-сервера Apache.

Отсутствие удобного графического интерфейса администратора – настройка Apache осуществляется путем редактирования его конфигурационного файла. Также большое количество параметров настройки может привести к возникновению уязвимостей. Возможны проблемы с производительностью на высоконагруженных сайтах.

64. Опишите архитектуру веб-сервера Apache.

Apache состоит из ядра и динамической модульной системы. Изменение параметров Apache происходит с помощью конфигурационных файлов.

65. Опишите функции ядра веб-сервера Apache.

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, работа с протоколом HTTP и система загрузки модулей.

66. Опишите конфигурацию веб-сервера Apache.

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации:

• Конфигурация сервера

• Конфигурация виртуального хоста

• Конфигурация уровня каталога

67. Что такое URI, URL и чем они различаются.

URI - Uniform Resource Identifier (унифицированный идентификатор ресурса)

URL - Uniform Resource Locator (унифицированный определитель местонахождения ресурса)

URN - Unifrorm Resource Name (унифицированное имя ресурса)

URI по сути является последовательностью символов, которая идентифицирует какой-то ресурс.

URL — это URI, который помимо идентификации ресурса предоставляет ещё и информацию о местонахождении этого ресурса.

68. Что такое HTTP-запрос?

HTTP запросы – это сообщения, отправляемые клиентом, чтобы инициировать реакцию со стороны сервера.

69. Опишите существующие HTTP-запросы.

GET – запрос содержимого ресурса

POST – передача данных ресурсу

PUT – загрузка содержимого запроса на ресурс

PATCH – частичное изменение ресурса

DELETE – удаление ресурса

Также существуют HEAD, OPTIONS, CONNECT, TRACE

70. Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса?

Для обработки запросов используются суперглобальные переменные, содержащие данные о запросе и параметрах (напр., $_GET и $_POST).

71. Опишите создание HTML-форм на PHP.

Это обычная форма HTML без каких-либо специальных тегов. Каждый элемент формы автоматически становится доступным программам на PHP. Когда пользователь заполнит форму и нажмёт кнопку отправки, будет вызвана страница, указанная в свойстве action. Используется суперглобальная переменная $_POST, которая содержит все POST-данные. Например, $_POST['name'] содержит данные из поля ‘name’формы.

72. Опишите API как средство интеграции приложений.

API в рамках Интернета нужно для того, чтобы использовать уже существующую функциональность в веб-приложениях без переделывания готовых успешных решений.

73. Что такое Web API?

API для веб-сервера.

74. Приведите пример API.

API GitHub, API Spotify, API VK

75. Что такое REST?

REST (Representational State Transfer) — это набор правил того, как программисту организовать написание кода приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать

76. Как организована передача данных в архитектуре REST?

- Применяется модель клиент-сервер
- Используется механизм кэширования
- Отсутствие промежуточного состояния и отсутствие хранения клиентского состояния на сервере
- Наличие унифицированного интерфейса
- Распределение архитектуры на слои

- Клиенты изменяют состояние системы только через действия, опредленные на сервере

77. Как организована работа REST?

Работа REST организована через обращение к ресурсам по уникальным URL. REST API взаимодействует при помощи HTTP запросов, выполняя стандартные функции CRUD.

78. Что такое SOAP?

Протокол SOAP (Simple Object Access Protocol) − набор правил для обмена структурированными сообщениями в распределенной вычислительной среде.

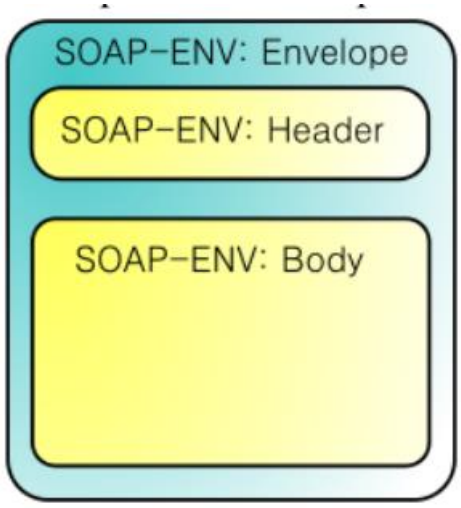
79. Чем SOAP отличается от REST?

REST был создан для решения проблем SOAP. Он состоит только из простых рекомендаций и позволяет разработчикам реализовывать рекомендации по-своему. Он допускает различные форматы сообщений, такие как HTML, JSON, XML и простой текст, в то время как SOAP допускает только XML. REST также является более легкой архитектурой, поэтому веб-сервисы RESTful имеют более высокую производительность.

80. Для чего нужен SOAP-процессор?

В общем случае это обработчик на сервере, отвечающий за обработку SOAP-сообщения в XML формате.

81. Опишите общую структуру SOAP-сообщения.



Fault – необязательный элемент, который предоставляет информацию об ошибках, которые произошли при обработке сообщений.

82. Что такое и что содержит Конверт (SOAP Envelope)?

Envelope – корневой элемент, который определяет сообщение и пространство имен, используемое в документе.

83. Что такое и что содержит Заголовок SOAP (SOAP Header)?

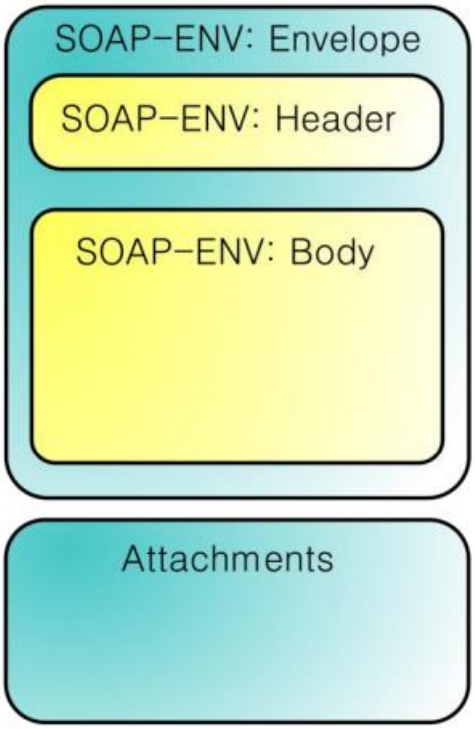
Header – содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации.

84. Что такое и что содержит Тело SOAP (SOAP Body)?

Body – содержит сообщение, которым обмениваются приложения.

85. Опишите SOAP-сообщение с вложением.

В SOAP возможна передача данных нетекстового типа, рассматриваемых в двоичном виде, – вложения.



86. Что такое graphql?

GraphQL — это язык запросов и манипулирования данными для API с открытым исходным кодом. Рядом преимуществ данной технологии является:

- Позволяет клиенту точно указать, какие данные ему нужны.

- Облегчает агрегацию данных из нескольких источников.
- Использует систему типов для описания данных.

87. Что такое Распознаватели (resolvers) в graphql?

Распознаватели используются для получения данных: для каждого типа данных создается свой распознаватель, который «знает», как и где получить объект этого типа.

88. Из чего состоит экосистема graphql, что нужно, чтобы использовать данную технологию?

Основой GraphQL являются документы, схема и распознаватели. Документом называется отправляемый и обрабатываемый клиентом файл, содержащий операции и фрагменты. Для описания данных, которые могут быть получены клиентом от сервера, используется схема. Распознаватели используются для получения данных: для каждого типа данных создается свой распознаватель, который «знает», как и где получить объект этого типа.

89. Что такое валидация данных и для чего она нужна?

Валидация данных – процесс проверки данных различных типов по критериям корректности и полезности для конкретного применения. Валидация нужна для предотвращения сбоя системы и для избежания дополнительной нагрузки на систему.

90. Где и когда выполнять валидацию данных?

Желательно валидировать данные как можно раньше: это упрощает код и снижает нагрузку на центральные узлы системы при распределённом сборе.

91. Как выполнять валидацию данных?

Наибольшее практическое применение находят методы, которые можно применить сразу в момент ввода данных в систему: проверки типа данных, простая проверка диапазона и ограничений, проверка кода и перекрестных ссылок.

92. Приведите пример с поэтапной валидацией данных.

Проверка данных формы на клиенте, проверка в коде сервера, проверка при занесении в базу данных.

93. Что такое запрос и мутация в graphql и чем они отличаются?

Это типы операций, прописанные в схеме. Запрос – возвращает запрошенные клиентом данные. Мутация – производит манипуляции с данными и возвращает клиенту измененные данные. Хотя технически любая операция может быть реализована так, чтобы перезаписать данные, по соглашению предполагается, что изменения должны быть отправлены явным образом через мутации. Также поля запроса выполняются параллельно, а мутации – последовательно.